Equilibrium

A zero-indexed array A consisting of N integers is given. An equilibrium index of this array is any integer P such that 0 ≤ P < N and the sum of elements of lower indices is equal to the sum of elements of higher indices, i.e.

A[0] + A[1] + ... + A[P−1] = A[P+1] + ... + A[N−2] + A[N−1].

Sum of zero elements is assumed to be equal to 0. This can happen if P = 0 or if P = N−1.

For example, consider the following array A consisting of N = 8 elements:

A[0] = -1
A[1] = 3
A[2] = -4
A[3] = 5
A[4] = 1
A[5] = -6
A[6] = 2
A[7] = 1

P = 1 is an equilibrium index of this array, because:

A[0] + A[1] + A[2] = −2 = A[4] + A[5] + A[6] + A[7]

P = 7 is also an equilibrium index, because:

A[0] + A[1] + A[2] + A[3] + A[4] + A[5] + A[6] = 0

and there are no elements with indices greater than 7.

P = 8 is not an equilibrium index, because it does not fulfil the condition 0 ≤ P < N.

Write a function:

```
object Solution { def solution(a: Array[Int]): Int }
```

that, given a zero-indexed array A consisting of N integers, returns any of its equilibrium indices. The function should return −1 if no equilibrium index exists.

For example, given array A shown above, the function may return 1, 3 or 7, as explained above.

Assume that:

- N is an integer within the range [0..100,000]
- each element of array A is an integer within the range [−2,147,483,648..2,147,483,647]

Complexity:

- expected worst-case time complexity is O(N)
- expected worst-case space complexity is O(N), beyond input storage (not counting the storage required for input arguments)

Elements of input array can be modified.